The *Entropy* Mechanism of Reinforcement Learning for Reasoning Language Models

Ganqu Cui¹*, Yuchen Zhang^{1,4}*, Jiacheng Chen¹*, Lifan Yuan³, Zhi Wang⁵, Yuxin Zuo², Haozhan Li², Yuchen Fan¹, Huayu Chen², Weize Chen², Zhiyuan Liu², Hao Peng³, Lei Bai¹, Wanli Ouyang¹, Yu Cheng^{1,6}†, Bowen Zhou^{1,2}†, Ning Ding^{2,1}†

¹ Shanghai AI Laboratory ² Tsinghua University ³ UIUC ⁴ Peking University ⁵ Nanjing University ⁶ CUHK

Code: https://github.com/PRIME-RL/Entropy-Mechanism-of-RL

Abstract

This paper aims to overcome a major obstacle in scaling reinforcement learning (RL) for reasoning with large language models (LLMs), namely the collapse of policy *entropy*. Such phenomenon is consistently observed across vast RL runs without entropy intervention, where the policy entropy dropped sharply at the early training stage, leading to an overly confident policy model. As a consequence, this diminished exploratory ability is always accompanied with the saturation of policy performance. In practice, we establish a transformation equation $R=-a\exp\mathcal{H}+b$, between entropy \mathcal{H} and downstream performance R, where a, b are fitting coefficients. This empirical law strongly indicates that, the policy performance is traded from policy entropy, thus bottlenecked by its exhaustion, and the ceiling is fully predictable ($\mathcal{H}=0, R=-a+b$). Our finding necessitates entropy management for continuous exploration toward scaling compute for RL. To this end, we investigate entropy dynamics both theoretically and empirically. Our derivation highlights that, the change in policy entropy is driven by the covariance between action probability and the change in logits, which is proportional to its advantage when using Policy Gradient-like algorithms (Williams, 1992). This is to say, a high-probability action with high advantage would reduce policy entropy, while a rare action with high advantage would increase policy entropy. Empirical study shows that, the values of covariance term and entropy differences matched exactly, supporting the theoretical conclusion. Moreover, the covariance term stays mostly positive throughout training, further explaining why policy entropy would decrease monotonically. Through understanding the mechanism behind entropy dynamics, we motivate to control entropy by restricting the update of high-covariance tokens. Specifically, we propose two simple yet effective techniques, namely Clip-Cov and KL-Cov, which clip and apply KL penalty to tokens with high covariances respectively. Experiments show that these methods encourage exploration, thus helping policy escape entropy collapse and achieve better downstream performance.

"Nature never undertakes any change unless her interests are served by an increase in entropy."

—Max Planck

^{*}Equal contribution.

[†]Corresponding Authors.

Contents

1	Intr	roduction	3					
2	The	Predictable "Collapse" of Policy Entropy	4					
	2.1	Preliminaries	4					
	2.2	Settings	5					
	2.3	A First Glance: Entropy Collapse and Performance Saturation	5					
	2.4	Fitting the Curves between Entropy and Performance	6					
	2.5	Understanding the Coefficients	7					
	2.6	Discussion	7					
3	Dyn	namics Analysis of Policy Entropy	8					
	3.1	Entropy Dynamics of Softmax Policy	8					
	3.2	Entropy Dynamics under Policy Gradient / Natural Policy Gradient Algorithms	9					
	3.3	Empirical Verification	9					
4	Entropy Control by Covariance Regularization							
	4.1	Effect of Entropy Regularization	10					
	4.2	Suppressing Tokens with High Covariances	11					
	4.3	Experiments	13					
	4.4	Get Policy Entropy Controlled	14					
	4.5	Discussion	15					
5	Rela	ated Work	15					
6	Con	nclusion	15					
A	Trai	Training Details for Different Models						
В	B More Fitting Results							
C	Fitti	Fitting Results of Training with Different Dataset.						
D	Fitti	Fitting Results of Instruct Models						
E	Pro	of	21					
	E.1	Useful Lemmas	21					
	E.2	Proof for Lemma 1	21					
	E.3	Proof for Proposition 1	22					
	E.4	Proof for Theorem 2	22					

1 Introduction

Applied to recent reasoning-centric large language models (LLMs), reinforcement learning (RL) escapes narrow, task-specific confines: the models' sweeping generalization introduces a new axis that vastly enlarges the exploratory landscape. This shift has yielded impressive reasoning gains (OpenAI, 2024a; DeepSeek-AI et al., 2025), yet the dilemma persists—scaling training compute for *learning from experience* (reinforcement learning) (Silver & Sutton, 2025) rather than *imitation learning* (pre-training and finetuning) remains non-trivial. Among the challenges emerges a major obstacle, the diminishment of *policy entropy*.

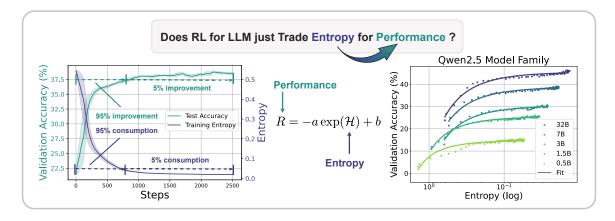


Figure 1: *Left:* Entropy collapse and performance saturation. Over 95% entropy drop/performance gains take place at the early stage of RL training. The model then reaches a plateau with little improvement. *Right:* The predictable relationship between validation performance and policy entropy. Without intervention, the policy "trades" entropy for performance exponentially, showing clear ceilings that hinder further policy enhancement.

The core challenge in RL is the exploitation-exploration trade-off (Sutton, 1988), balancing the reuse of proven strategies against the search for novel ones. For exploration, a key concept quantifying the exploratory potential is *policy entropy*, which measures the uncertainty in the policy's action selection process. In RL literature, the ability to mitigate the decline of policy entropy is regarded as essential to most algorithms (Williams & Peng, 1991; Williams, 1992; Eysenbach & Levine, 2021), and policy entropy has been intensively steered and actively controlled via regularization (Ziebart et al., 2008; Schulman et al., 2017b; Haarnoja et al., 2018).

For LLMs, while the typical behavior of policy entropy remains largely understudied (Yu et al., 2025; He et al., 2025), we find an intriguing and consistent pattern from broad experiments: Policy entropy sharply declines to near 0 in a few training steps, demonstrating that the policy becomes extremely certain. Consequently, the inability to explore new paths leads to a performance plateau, where the validation performance also struggles to improve at the same time. Quantitatively, we further reveal that, without entropy intervention like entropy loss or KL regularization, the downstream performance is fully predictable from policy entropy, and the fitted curve is a simple exponential function $R = -a \exp \mathcal{H} + b$, as shown in Figure 1. Basically, the policy is trading uncertainty (entropy) for rewards (Yue et al., 2025) in a predictable manner.

This empirical law results in two critical corollaries: (1) Like Scaling Laws (Kaplan et al., 2020; Hoffmann et al., 2022), the exploitation-exploration curve is predetermined given the policy model and training data. This allows us to predict policy performance at the early stage of RL and predict the performance of large models given small models (OpenAI, 2024b) (Sec. 2.4). (2) More importantly, this equation indicates that the upper bound of the policy performance is also deterministic with the exhaustion of policy entropy $(\mathcal{H}=0,R=-a+b)$, so the return of scaling training compute for RL could be marginal. What's worse, naively applying entropy regularization methods are proven ineffective (Sec. 4.1). In short, scalable RL calls for breaking the entropy bottleneck.

Solving this issue requires principled understandings of the mechanisms behind this observation, i.e., why policy entropy decreases monotonically? To this end, we analyze the dynamics of policy entropy both theoretically and empirically. Our key findings highlight that, for softmax policy like LLMs, the entropy change between two consecutive steps is proportional to the covariance of the log-probability and corresponding logit change for an action (Liu, 2025). Furthermore, under Policy Gradient (Williams, 1992)-like and Natural Policy Gradient (Kakade, 2001)-like algorithms, the logit difference is proportional to the action advantage.

Intuitively, an action with high advantage and high probability would reduce policy entropy, while a rare action with a high advantage would increase entropy. This theoretical conclusion is validated by experimental results. At the early stage, the policy demonstrates high covariance on training data, implicating the policy's confidence is well-calibrated (Kadavath et al., 2022), thus can safely exploit trajectories with high confidence, strengthening belief and minimize entropy (Zuo et al., 2025; Zhang et al., 2025; Agarwal et al., 2025). As training progresses, the covariance gradually declines but still maintains positive, continually dragging policy entropy even lower.

The analysis of entropy dynamics demonstrates that, the high covariance is detrimental to scalable RL, which provides us guidelines about uplifting policy entropy—limit the step sizes of high-covariance tokens. We thereby motivate to design two corresponding strategies aiming at entropy control, namely Clip-Cov and KL-Cov, to replace the clip and PPO-KL methods in surrogate loss (Schulman et al., 2017b). Clip-Cov randomly selects a small portion of tokens with positive covariances and detach their gradients. KL-Cov, on the other hand, applies KL penalty on tokens with the largest covariances. Experiment results show that, we can actively control policy entropy by tuning threshold parameters. Consequently, the policy model escapes the low entropy trap and achieves better performance on mathematical reasoning.

It has become a common belief that the training computes will shift from pre-training to post-training, especially RL (Silver & Sutton, 2025). On the road to scale RL with increased compute, it is pivotal to keep exploration, find novel paths, and continuously improve, thus utilizing the compute even better (Sutton, 2019). Our work provides an entropy-centric perspective for this purpose, which hopefully could be beneficial for understanding, analyzing, and advancing the underlying mechanisms of RL for LLMs.

2 The Predictable "Collapse" of Policy Entropy

TAKEAWAY

Without intervention, e.g., entropy or KL regularization, policy entropy is *traded for reward predictably* during RL. The empirical quantitative relationship between validation reward R and policy entropy \mathcal{H} can be expressed as $R = -a \exp(\mathcal{H} + b)$. Then:

- It suggests the predictability of policy performance from entropy;
- The coefficients a, b reflect internal characteristics of policy and data;
- The performance ceiling of the policy ($\mathcal{H} = 0$, R = -a + b) greatly burdens the scalability of RL for LLM reasoning.

In this section, we manage to answer the research question: What is the typical behavior of policy entropy during RL for LLMs? Through extensive experiments, we observe a consistent "entropy collapse" phenomenon, which is not favored in RL since the policy would struggle to explore new paths (Sec. 2.3). We further extend it to an empirically predictable relation between policy entropy and validation performance (Sec. 2.4), and analyze the coefficients in the equation (Sec. 2.5).

2.1 Preliminaries

We consider tuning LLMs with RL on verifiable tasks, such as math and coding, to avoid reward hacking. Given an input prompt x, an LLM π_{θ} autoregressively generates an output sequence y, which consists of T tokens $\{y_1, \cdots, y_t, \cdots, y_T\}$. We use RL to optimize the LLM policy to maximize the cumulative rewards r received from the verifier:

$$\max_{\theta} J(\theta) := \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}, \boldsymbol{y} \sim \pi_{\theta}(\boldsymbol{x})} [r(\boldsymbol{y})]$$
 (1)

where \mathcal{D} is the training distribution.

To optimize the objective function, it is a common practice to use the Policy Gradient algorithm (Williams, 1992) for gradient estimation:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\boldsymbol{x} \sim \mathcal{D}, \boldsymbol{y} \sim \pi_{\theta}(\boldsymbol{x})} \left[\sum_{t=0}^{T} \nabla_{\theta} \log \pi_{\theta}(y_{t} | \boldsymbol{y}_{< t}) A_{t} \right].$$
 (2)

Here A_t is the advantage of current action and is implemented differently across RL algorithms. If we only have rewards for the full trajectory, the vanilla REINFORCE algorithm (Williams, 1992) directly defines

 $A_t = r(y)$. To reduce variance, GRPO (Shao et al., 2024) and RLOO (Kool et al., 2019; Ahmadian et al., 2024) further incorporates group-wise normalization. For example, GRPO samples K responses for each prompt and estimates the advantage as follows:

$$A_{t} = \frac{r(\boldsymbol{y}) - \operatorname{mean}\left(r\left(\boldsymbol{y^{1:K}}\right)\right)}{\operatorname{std}\left(r\left(\boldsymbol{y^{1:K}}\right)\right)}.$$
(3)

To handle off-policy data and constrain the policy update size, PPO (Schulman et al., 2017b) proposed to optimize a surrogate loss.

$$L(\theta) = \mathbb{E}_t \left[\min \left(\frac{\pi_{\theta}(y_t | \boldsymbol{y}_{< t})}{\pi_{\theta_{\text{old}}}(y_t | \boldsymbol{y}_{< t})} A_t, \text{clip}\left(\frac{\pi_{\theta}(y_t | \boldsymbol{y}_{< t})}{\pi_{\theta_{\text{old}}}(y_t | \boldsymbol{y}_{< t})}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \right]$$
(4)

Policy entropy. Policy entropy quantifies the predictability or randomness inherent in the actions selected by an agent. Given policy model π_{θ} , training dataset \mathcal{D} , we measure the average token-level entropy of the policy model on training data, which is defined as follows:

$$\mathcal{H}(\pi_{\theta}, \mathcal{D}) = -\mathbb{E}_{\mathcal{D}, \pi_{\theta}} \left[\log \pi_{\theta}(y_t | \boldsymbol{y}_{< t}) \right] = -\frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \frac{1}{|\boldsymbol{y}|} \sum_{t=1}^{|\boldsymbol{y}|} \mathbb{E}_{y_t \sim \pi_{\theta}} \left[\log \pi_{\theta}(y_t | \boldsymbol{y}_{< t}, x) \right]$$
(5)

Such entropy quantifies the uncertainty level of the policy on current prompts and is widely adopted in maximum entropy RL as a regularization term (Ziebart et al., 2008; Haarnoja et al., 2017, 2018). In practice, we calculate the entropy for each batch of prompts randomly sampled from the training dataset.

2.2 Settings

We adopt a unified protocol covering 4 model families and 11 base models (0.5-32B parameters), verifiable task domains of math and coding evaluated on 8 public benchmarks, and 4 RL algorithms.

Models. The models adopted in our experiments span 4 model families and 11 widely used open-source base models. Specifically, these consist of the Qwen2.5 family (Qwen2.5-0.5B, 1.5B, 3B, 7B, 32B) (Qwen et al., 2025), the Mistral family (Mistral-7B-v0.3 (Jiang et al., 2023), Mistral-Nemo-Base-2407 (MistralAI-NeMo), Mistral-Small-3.1-24B-Base-2501 (MistralAI-Small-3)), the LLaMA family (LLaMA3.2-3B (Meta-Llama-3.2), LLaMA3.1-8B (Meta, 2024)), and DeepSeek-Math-7B-Base (Shao et al., 2024)).

Tasks and datasets. We primarily focus on math and coding problems with verifiable rewards. Due to inherent differences in the initial reasoning abilities between model families, we train models using data of different difficulty levels to stabilize the RL process. Details can be found in Appendix A. Meanwhile, we use the same data during downstream performance evaluation to maintain consistency. For math tasks, the evaluation datasets include MATH500 (Hendrycks et al., 2021), AIME 2024 (Li et al., 2024), AMC (Li et al., 2024), OlympiadBench (He et al., 2024), and OMNI-MATH (Gao et al., 2024). For code tasks, we split the testset of Eurus-2-RL-Code (Cui et al., 2025) and KodCode (Xu et al., 2025).

RL training. We start RL from the base models following the "Zero" setting (DeepSeek-AI et al., 2025) with the veRL framework (Sheng et al., 2024). For RL algorithms, we employ GRPO (Shao et al., 2024), REINFORCE++ (Hu, 2025), and PRIME (Cui et al., 2025). For hyperparameters, we use a learning rate of 5×10^{-7} for the policy model and 10^{-6} for the implicit PRM (Yuan et al., 2025) in PRIME. Both policy and PRMs use a batch size of 256 and a micro-batch size of 128. The rollout stage collects 512 prompts with 8 sampled responses. By default, we set the reference KL divergence coefficient to 0. The ϵ in policy loss (Equation 4) is 0.2. We filter out prompts that receive all correct or incorrect responses.

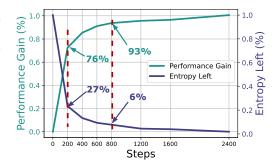


Figure 2: Avg. entropy consumption/performance gain (%) in 11 RL runs with different models.

2.3 A First Glance: Entropy Collapse and Performance Saturation

Across all experiments, we observe a consistent pattern: The policy entropy encounters a sharp drop at the very beginning of the training, and it keeps declining monotonically to near zero. Meanwhile, the policy's

validation performance presents an inverse trend, where it rises rapidly when training starts, and then saturates at a certain level.

Figure 2 illustrates the average normalized entropy consumption/performance gain in percentage throughout 2400-gradient step RL runs with 11 different models. We can see that 73% of the entropy consumption and 76% of the performance gain occurred in just the first 200 gradient steps (1/12 of training), and the first 800 (1/3) steps account for over 93% performance gains together with 94% entropy losses. This means that over 2/3 of the training steps yielded marginal returns.

2.4 Fitting the Curves between Entropy and Performance

Motivated by the observed entropy collapse phenomenon, we conduct a more detailed quantitative analysis. Through extensive experiments, we find the downstream performance (accuracy) and entropy can be fitted in the exponential function:

$$R = -a\exp(\mathcal{H}) + b,\tag{6}$$

where R represents the validation performance and \mathcal{H} is entropy. The fitting results of different model families with GRPO are presented in Figure 3 and 4. It is worth noting that, the fitted curves precisely describe the performance-entropy relationships over all conducted experiments, with models spanning all kinds of sizes, families, and different tasks. Only 2 coefficients are needed for fitting the curve of over 200 data points, showing a high degree of regularity. The fitting results of instruct models and training on different datasets can be found in Appendix B.

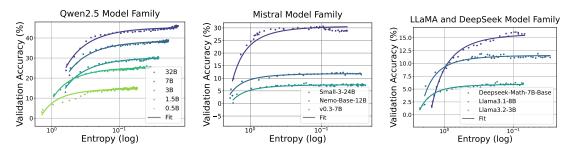


Figure 3: Fitting curves between policy entropy and validation performance on math task. We conduct validation every 4 rollout steps until convergence.

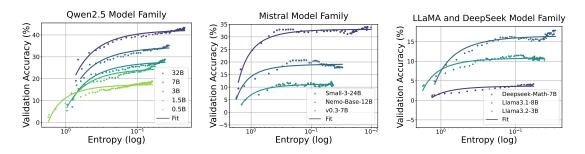
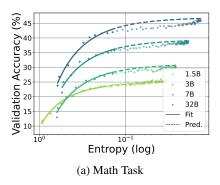


Figure 4: Fitting curves between policy entropy and validation performance in coding task. We conduct validation every 4 rollout steps until convergence.

Predicting late stage from early stage. As we can precisely fit a curve between policy entropy and validation performance, one straightforward application of this fitting is to predict policy performance at low entropy with observations from high entropy data points. To verify that the functional form can be applied at the early stage of RL training, we take a step further by fitting the function within limited training steps and using the fitted function to predict the final performance.

Take Qwen2.5 family as an example, we fit the function form with coefficients a and b using only the first 36 training steps. Using this function, we perform an advance prediction for the subsequent 200 training steps.



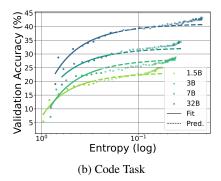


Figure 5: Predicting the final performance of Qwen2.5 family with only 15% training steps with the fitted function. The average RMSE is 0.9% and 1.2% for all predicted steps, 0.5% and 1.9% for final step performance, respectively.

As shown in Figure 5, for the math and coding task, we achieve an average Root Mean Square Error (RMSE) of 0.9% and 1.2% during prediction, 0.5% and 1.9% for final performance, respectively. It suggests that the late stage performance of the policy can be estimated early in training, without the need to run the full RL process. Moreover, we can also obtain the final performance of the policy when it becomes static. With $\mathcal{H}=0$, R=-a+b, which is the upper bound of the policy given the training data.

2.5 Understanding the Coefficients

The coefficients are algorithm-irrelevant. We investigate whether different RL algorithms would affect the fitted function. Figure 6 plots the fitted curves with GRPO, RLOO, and PRIME. We find that, although these algorithms apply distinct advantage estimation methods, they do not influence the fitted entropy-performance function. This indicates that the coefficients a,b reflect some intrinsic properties of the policy model and training data.

Predicting coefficients when scaling parameters. Taking a closer look at the coefficients a,b, their meanings are clear. By differentiating the equation, we derive $dR/d\mathcal{H} = -a\exp(\mathcal{H})$, which means a is the rate at which the model converts entropy into downstream performance. Also, as stated above, -a+b is the maximum validation score the model can achieve when entropy is fully depleted. Intuitively, a,b should be relevant with model sizes, where larger models could trade entropy for reward more efficiently, as well as achieve higher performance.

To validate this, we again adopt Qwen2.5 model family, since they have similar architecture and undergo similar training process. In Figure 7, we plot the model parameter count (without embedding) versus a,b on math and coding tasks. It is observed that, both a and b vary smoothly with policy size at a log-linear rate. This log-linear relationship between model sizes and coef-

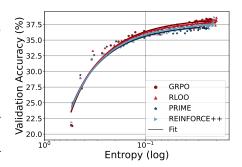


Figure 6: Training Qwen2.5-7B with different RL algorithms.

ficients is also observed in Gao et al. (2022). It allows us to extrapolate the coefficients of larger models based on the training dynamics of smaller models, extending the predictability to the dimension of model sizes. In other words, it enables us to predict the final performance of larger LMs through RL training without actually training them, once we train smaller models within the same family and get their coefficients. Figure 13 also illustrates that the coefficients are related with training data.

2.6 Discussion

The predictability. To now, we have established predictability between (1) policy performance and entropy; (2) coefficients in (1) and model sizes. Such predictability reminds us of Scaling Laws for language models (Kaplan et al., 2020; Hoffmann et al., 2022) and RLHF (Gao et al., 2022). It seems that, RL with LLMs keeps trading entropy for reward throughout training. However, other works that adopt different policy models (Luo et al., 2025) or use off-policy data (Yan et al., 2025) observed distinct entropy patterns. Therefore, this predictability

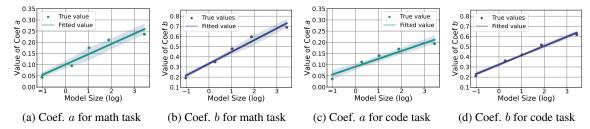


Figure 7: Fitted curves between coefficients and model sizes of Qwen2.5 model family. The model sizes are parameter counts (B) without embeddings. a, b are obtained from experiments in Sec. 2.4. We use log-linear function to fit the curve.

is not arguably universal, and we call for a more in-depth analysis of the entropy behavior under different conditions.

The ceiling. There is an intensive discussion questioning whether RL merely elicits the latent behaviors that were already learned in pre-training, thus cannot break the ceiling of the base model (Yue et al., 2025). Our results conditionally support this claim that, if policy entropy diminishes, the ceiling not only exists, but also can be predicted. However, we argue that it is not the intrinsic limitation of RL that sets up the ceiling, but the entropy mechanism of LLMs leads to the result. Although LLMs provide us with strong policy priors, their output distributions are also narrowed, which might hinder the exploration potential during RL.

3 Dynamics Analysis of Policy Entropy

TAKEAWAY

- (1) For softmax policy including LLMs, the change of policy entropy is determined by the **covariance** between the log-probability and the change in logits of actions.
- (2) For Policy Gradient and Natural Policy Gradient, the change in logits is proportional to the action advantage, meaning that a high covariance leads to a quick decrease of policy entropy, as observed in RL for LLM reasoning.

We have unveiled that the entropy collapse issue will greatly obstacle RL scaling for LLM reasoning. To solve it, we need a more principled understanding of the *dynamics of policy entropy*, *i.e.*, when will entropy decrease and when will entropy increase. In this section, we focus on the entropy dynamics, especially the step-wise entropy difference $\mathcal{H}(\pi_{\theta}^{k+1}) - \mathcal{H}(\pi_{\theta}^{k})$. We start from a theoretical perspective, firstly derive the first-order derivative of entropy for softmax policy in Sec. 3.1, then extend further in Policy Gradient and Natural Policy Gradient algorithms (Sec. 3.2). Afterwards, we validate our conclusion with experiments (Sec. 3.3).

3.1 Entropy Dynamics of Softmax Policy

For step k, we try to calculate the entropy difference before and after one step parameter update, i.e., $\mathcal{H}(\pi_{\theta}^{k+1})$ and $\mathcal{H}(\pi_{\theta}^{k})$. For this, we first consider an intrinsic property of LLMs that they are softmax policies, which means the policies are parameterized by

$$\pi_{\theta}(a|s) = \frac{\exp(z_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(z_{s,a'})}.$$
(7)

Here $s \sim d_{\pi_{\theta}}$ and $a \sim \pi_{\theta}^{k}(\cdot|s)$ represent state and action, $z_{s,a}$ is the output logit of action a given state s. For any softmax policy, we have the following Lemma:

Lemma 1 (Entropy difference of softmax policy) (Proof in Appendix E.2, adapted from Liu (2025)) Assume that policy π_{θ} is a tabular softmax policy, where each state-action pair (s,a) is associated with an individual logit parameter $z_{s,a} = \theta_{s,a}$, the difference of policy entropy given state s between two consecutive steps under first-order approximation satisfies

$$\mathcal{H}(\boldsymbol{\pi}_{\theta}^{k+1}) - \mathcal{H}(\boldsymbol{\pi}_{\theta}^{k}) \approx \mathbb{E}_{s \sim d_{\boldsymbol{\pi}_{\theta}}} \left[\mathcal{H}(\boldsymbol{\pi}_{\theta}^{k+1}|s) - \mathcal{H}(\boldsymbol{\pi}_{\theta}^{k}|s) \right] \approx \mathbb{E}_{s \sim d_{\boldsymbol{\pi}_{\theta}}} \left[-Cov_{a \sim \boldsymbol{\pi}_{\theta}^{k}(\cdot|s)} \left(\log \boldsymbol{\pi}_{\theta}^{k}(a|s), \ z_{s,a}^{k+1} - z_{s,a}^{k} \right) \right]$$

Here $z_{s,a}^{k+1} - z_{s,a}^k$ is the change in the output logits between step k and step k+1. This Lemma indicates that, the change of policy entropy approximately equals the negative covariance between log-probability of the action and the change of logits. That is to say, when an action a receives a high probability from the policy before updating, and its corresponding logit is also increasing after updating, then it will decrease the policy entropy.

3.2 Entropy Dynamics under Policy Gradient / Natural Policy Gradient Algorithms

From Lemma 1, the step-wise difference of output logits $z_{s,a}^{k+1} - z_{s,a}^{k}$ contributes to change of entropy, which is related with the specific training algorithm in use. Here, we further derive the logits change under Policy Gradient (Williams, 1992) and Natural Policy Gradient (Kakade, 2001) algorithms.

Assuming that we are updating the actor policy via Policy Gradient, then $z_{s,a}^{k+1} - z_{s,a}^k = -\eta \cdot \nabla_z J(\theta)$, where $J(\theta)$ denotes the objective function and η denote the learning rate. $\nabla_z J(\theta)$ is calculated with Eq. 2, we have the following proposition:

Proposition 1 (Difference of policy logits in vanilla policy gradient) (Proof in Appendix E.3) Let the actor policy π_{θ} be a tabular softmax policy and updated using Eq. 2 via gradient backtracking with learning rate η , the difference of $z_{s,a}$ between two consecutive steps satisfies

$$z_{s,a}^{k+1} - z_{s,a}^{k} = \eta \, \pi_{\theta}(a \mid s) \, A(s,a)$$

Applying Proposition 1 to Lemma 1, we can further describe entropy change with the following theorem:

Theorem 1 (Entropy change under policy gradient) *Let the actor policy* π_{θ} *be a tabular softmax policy, and* π_{θ} *be updated via vanilla policy gradient, the difference of policy entropy given state s between two consecutive steps satisfies*

$$\mathcal{H}(\pi_{\theta}^{k+1}|s) - \mathcal{H}(\pi_{\theta}^{k}|s) \approx -\eta \cdot Cov_{a \sim \pi_{\alpha}^{k}(\cdot|s)} \left(\log \pi_{\theta}^{k}(a|s), \pi_{\theta}^{k}(a|s) \cdot A(s,a)\right)$$

Theorem 1 reveals how policy entropy changes under the policy gradient method. Intuitively, an action a receives both high/low probability and high/low advantage would lower the entropy, and vice versa. At the early stage, the policy demonstrates high covariance on training data, implicating the policy's confidence is well-calibrated (Kadavath et al., 2022), thus can safely exploit trajectories with high confidence, strengthening belief and minimize entropy (Zuo et al., 2025; Zhang et al., 2025; Agarwal et al., 2025).

Liu (2025) conducted derivation for Natural Policy Gradient. We present the conclusion below.

Theorem 2 (Entropy change under natural policy gradient) (Proof in Appendix E.4) Let the actor policy π_{θ} be a tabular softmax policy, and π_{θ} is updated via natural policy gradient (Kakade, 2001), the difference of policy entropy given state s between two consecutive steps satisfies

$$\mathcal{H}(\pi_{\theta}^{k+1}|s) - \mathcal{H}(\pi_{\theta}^{k}|s) \approx -\eta \cdot \textit{Cov}_{a \sim \pi_{\theta}^{k}(\cdot|s)} \left(\log \pi_{\theta}^{k}(a|s) \;, A(s,a)\right)$$

Conclusion. From Theorem 1 and Theorem 2, we obtain the intuitive insight that, in principle, a strong positive correlation between the action probability P(a) under the current policy and the corresponding advantage value A(a), on average, leads to a decrease in policy entropy. Conversely, a negative correlation tends to increase the entropy. This deeper understanding of the dynamics of policy entropy provides a theoretical foundation for designing practical strategies for entropy control in policy optimization.

3.3 Empirical Verification

The preceding theoretical analysis provides insights about the factors influencing policy entropy when optimizing a softmax policy via a policy gradient algorithm. In this section, we conduct experiments to validate the theoretical conclusion, specifically, Theorem 1.

Settings. We apply GRPO with policy gradient, *i.e.* on-policy learning without PPO surrogate loss, on Qwen2.5-7B. In this context, we adopt the bandit setting where the prompt x is the state, and whole response y is the action. Then the covariance term becomes:

$$Cov_{a \sim \pi_{\theta}(\cdot \mid s)} \left(\log \pi_{\theta}(a \mid s), \pi_{\theta}(a \mid s) \cdot A(s, a) \right) = Cov_{\boldsymbol{y} \sim \pi_{\theta}(\cdot \mid \boldsymbol{x})} \left(\log \pi_{\theta}(\boldsymbol{y} \mid \boldsymbol{x}), \pi_{\theta}(\boldsymbol{y} \mid \boldsymbol{x}) \cdot A(\boldsymbol{y}, \boldsymbol{x}) \right)$$
(8)

During training, we calculate the group-wise covariance for each prompt, and average across a batch of prompts. We further normalize the log-prob by the length of the response, which gives

$$\log \pi_{\theta}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{|\boldsymbol{y}|} \left[\sum_{t=1}^{|\boldsymbol{y}|} \log \pi_{\theta}(y_t \mid \boldsymbol{y}_{< t}, \boldsymbol{x}) \right]$$

$$0.0200$$

$$0.0175$$

$$0.0150$$

$$0.0125$$

$$0.0100$$

$$0.0025$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

$$0.0005$$

Figure 8: *Left*: The dynamics of policy entropy (step-wise entropy difference) and covariance during on-policy GRPO training. They show similar trends as expected from the theoretical results. *Right*: Different prompt groups show distinct covariance behaviors. Easier prompts with higher accuracy have higher covariances as well, while harder prompts have lower covariances.

Experiment results. We record two key metrics based on the aforementioned derivation $Cov(\cdot)$ and $\mathcal{H}(\pi_{\theta})$ across the training period and try to analyse their relationship and dynamics.

- 1) Similarity of dynamic between $Cov(\cdot)$ and $-d(\mathcal{H})$. According to Theorem 1, we have the theoretical result $-d(\mathcal{H}) \propto Cov(\cdot)$. As shown in LHS of Figure 8, the empirical curves of $-d(\mathcal{H})$ and $Cov(\cdot)$ exhibit highly similar dynamics, providing strong empirical support for the theorem. In particular, during the early stages of training, entropy \mathcal{H} decreases rapidly, accompanied by a relatively large and positive $Cov(\cdot)$. As the RL training progresses, the entropy decay slows down, and $Cov(\cdot)$ stabilizes at a lower level, reflecting the gradual convergence of the policy. It can also be observed that $Cov(\cdot)$ remains positive along the training process, thus resulting in a persistent decrease in entropy.
- 2) Variation in $Cov(\cdot)$ dynamics across examples of different difficulty. Leveraging our group-based sampling strategy, we categorize training examples by difficulty based on their accuracy. RHS of Figure 8 illustrates the covariance curves for three difficulty groups, where lower accuracy indicates higher difficulty. We observe that $Cov(\cdot)$ tends to be smaller in magnitude for harder examples, which aligns with intuition: when the model struggles to learn, high-probability actions are not reliably associated with higher expected returns. In contrast, for easier examples, where the model is more confident and better calibrated, $Cov(\cdot)$ tends to be higher, indicating a stronger alignment between action probabilities and advantage estimates.

4 Entropy Control by Covariance Regularization

TAKEAWAY

We can control policy entropy by **restricting the update of tokens with high covariances**, e.g., clipping (Clip-Cov) or applying KL penalty (KL-Cov). These simple techniques prevent policy from entropy collapse thus promoting exploration.

The analysis of entropy dynamics gives us guidelines for entropy control, regularizing the update step size of high-covariance actions. In this section, we introduce two simple yet effective techniques, KL-Cov and Clip-Cov, that control entropy precisely and achieve better downstream performance.

4.1 Effect of Entropy Regularization

A common approach in the RL literature to control policy entropy is to apply entropy loss (Schulman et al., 2017b). We conduct experiments to see whether it is effective for LLMs. Figure 9 presents the results of adding

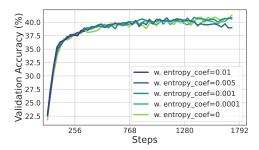
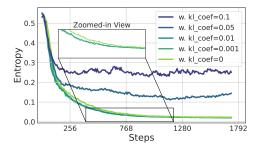


Figure 9: The policy entropy and validation accuracy of adding entropy loss where $L_{\text{ent}} = L - \alpha \mathcal{H}(\pi_{\theta})$. L is the original loss and α is the coefficient of entropy loss.



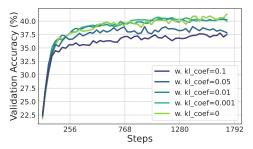


Figure 10: The policy entropy and validation accuracy of adding KL penalty between policy and reference model where $L_{\rm KL} = L + \beta \mathbb{D}_{\rm KL}(\pi_{\theta}||\pi_{\rm ref})$. L is the original loss and β is the coefficient of KL loss.

entropy loss. As demonstrated, entropy loss is highly sensitive to coefficients, while small coefficients have a minor influence on entropy (0.0001, 0.001), large coefficients lead to entropy explosion (0.01). Although setting the coefficient at 0.005 successfully stabilizes policy entropy, it does not outperform other baselines.

We also attempt to control the entropy by adjusting the KL penalty between the policy model and the reference model. In Figure 10, we report the results. Despite the reference KL achieves stable entropy values, it fails to improve policy and instead leads to a degradation in performance.

To summarize, naively adopting entropy regularization techniques from conventional RL struggles to solve the entropy bottleneck of LLMs. These regularization terms are either hyper-parameter sensitive (He et al., 2025) or degrade policy performance. Therefore, most recent works do not include them as well (Cui et al., 2025; Hu et al., 2025; Liu et al., 2025; Yu et al., 2025).

4.2 Suppressing Tokens with High Covariances

The unsuccessful attempt to incorporate entropy and reference KL regularization into the policy loss drives us to seek a more fundamental approach to control entropy. As previously elaborated, we know that the policy entropy dynamic is closely connected with the covariance between action probability and advantage. Meanwhile, as shown in Table 1, a small portion of tokens exhibit extremely high covariance, far exceeding the average. That is saying that these outlier tokens take a dominant part in triggering the entropy collapse. To mitigate their adverse effect, we aim to impose constraints on their contribution to the policy loss. In RL literature, two variants of PPO employ either clipping or KL penalty to constrain the policy updates (Schulman et al., 2017b), preventing overly aggressive changes. Drawing inspiration from these approaches, we propose two simple but effective covariance-aware methods Clip-Cov and KL-Cov to achieve this goal.

Table 1: Covariance distribution of Qwen2.5-7B in training step 1.

Group	Mean Value			
Top 0.02%	5.654			
Top 0.2%	3.112			
Top 2%	1.385			
Top 20%	0.351			
Top 50%	0.152			
All	0.003			

Natural policy gradient is rarely used in post-training of LLMs because of its time-consuming second-order optimization. But its introduction of target function with KL distance as constraint share a similar idea with

Table 2: Detailed results of GRPO, GRPO with clip-higher technique and our methods. For AIME and AMC,
the results are avg.@32. Bold denotes the best results.

Method	AIME24	AIME25	AMC	MATH-500	OMNI-MATH	OlympiadBench	Minerva	Avg.
Qwen2.5-7B								
GRPO	21.2	9.6	58.7	78.8	27.9	40.7	36.7	38.6
w. Clip-higher	18.1	11.5	56.6	79.2	29.8	43.3	40.4	38.8
w. CLIP-Cov	22.1	15.8	58.2	80.4	30.5	44.1	41.1	40.4
w. KL-Cov	22.6	12.9	61.4	80.8	29.1	42.6	38.2	40.6
GRPO	21.8	16.2	69.7	84.2	35.2	43.6	45.5	45.8
w. Clip-higher	35.6	22.3	69.5	77.2	35.1	42.5	43.0	47.2
w. CLIP-Cov	32.3	22.7	67.2	87.0	42.0	57.2	46.0	50.3
w. KL-Cov	36.8	30.8	74.5	84.6	39.1	49.0	46.3	52.2

TRPO (Schulman et al., 2015) and PPO. For this reason, we apply Theorem 2 into algorithms like PPO later in this section.

Supposing a batch of N rollout tokens, $\pi_{\theta}(y_i)$ denotes the output probability of the policy model for token y_i given its corresponding prefix. According to Theorem 2, we firstly define token-wise centered cross-product between log probability and advantage as:

$$Cov(y_i) = (\log \pi_{\theta}(y_i) - \frac{1}{N} \sum_{j=1}^{N} \log \pi_{\theta}(y_j)) \cdot (A(y_i) - \frac{1}{N} \sum_{j=1}^{N} A(y_j))$$
 (10)

The Cov is the covariance of each token in N. Its expectation is the covariance in Theorem 2.

Clip-Cov. In the Clip-Cov strategy, we clip a small fraction of high-covariance tokens out from policy gradient updates as follows. With Eq. 10 calculated, we randomly select $r \cdot N$ of high-covariance tokens according to the covariance value:

$$I_{\text{clip}} = I \sim \text{Uniform} (i \mid Cov(y_i) \in [\omega_{\text{low}}, \omega_{\text{high}}]\}, |r \cdot N|)$$
 (11)

Where I is short for index, r denotes the clip ratio. $\omega_{\text{low}}, \omega_{\text{high}}$ are two predefined bounds for covariance, respectively. Both are set much higher than the average covariance (>500×). Finally, tokens with the chosen indexes will be detached from the policy gradient, which is:

$$L_{\text{Clip-Cov}}(\theta) = \begin{cases} \mathbb{E}_t \left[\frac{\pi_{\theta}(y_t | \mathbf{y}_{\leq t})}{\pi_{\theta_{\text{old}}}(y_t | \mathbf{y}_{\leq t})} A_t \right], & t \notin I_{\text{clip}} \\ 0, & t \in I_{\text{clip}} \end{cases}$$
(12)

where the t is the t-th token in one rollout response and each t uniquely corresponds to a index i in N.

KL-Cov. The KL-Cov strategy is simpler. Specifically, similar to Clip-Cov, we first compute the covariance as in Eq. 10. Then, we rank and select tokens within the top-k proportion of covariance:

$$I_{KL} = \{ i \mid \text{Rank}(Cov(y_i)) \le k \cdot N \}, \tag{13}$$

The k here denotes the proportion of tokens that will be subjected to the KL penalty and $k \ll 1$. At last, we impose the KL penalty (KL divergence between the current policy and the rollout policy) on the selected tokens, the policy loss is computed as:

$$L_{\text{KL-Cov}}(\theta) = \begin{cases} \mathbb{E}_{t} \left[\frac{\pi_{\theta}(y_{t}|\boldsymbol{y}_{< t})}{\pi_{\theta_{\text{old}}}(y_{t}|\boldsymbol{y}_{< t})} A_{t} \right], & t \notin I_{\text{KL}} \\ \mathbb{E}_{t} \left[\frac{\pi_{\theta}(y_{t}|\boldsymbol{y}_{< t})}{\pi_{\theta_{\text{old}}}(y_{t}|\boldsymbol{y}_{< t})} A_{t} - \beta \mathbb{D}_{\text{KL}}(\pi_{\theta_{\text{old}}}(y_{t}|\boldsymbol{y}_{< t})||\pi_{\theta}(y_{t}|\boldsymbol{y}_{< t})) \right], & t \in I_{\text{KL}} \end{cases}$$

$$(14)$$

Where β is the coefficient to control the weight for the KL penalty. We present the pseudo-code in Listing 1.

```
def compute_policy_loss(old_log_prob, log_prob, advantages,
   select_ratio, method, **args):
   ratio = exp(log_prob - old_log_prob)
   pg_losses1 = -ratio * advantages
    # calculate token wise centered cross - product
    covs = (log_prob - log_prob.mean()) * (advantages - advantages.mean
   ())
   select_num = int(select_ratio * len(pg_losses1))
    if method == "clip_cov":
        pg_losses2 = -clip(ratio, args["clip_range_lb"], args["
           clip_range_ub"]) * advantages
        # randomly select index to be detached
        clip_idx = random_select(covs[covs > args["cov_lb"] & covs <</pre>
   args["cov_ub"]], num=select_num)
        pg_losses1[clip_idx].detach_()
        pg_losses2[clip_idx].detach_()
        pg_loss = maximum(pg_losses1, pg_losses2).mean()
    if method == "kl_cov":
        kl_coef = args["kl_coef"]
        kl_penalty = (log_prob - old_log_prob).abs()
        pg_losses = pg_losses1 + kl_coef * kl_penalty
        # find out index with highest conviriance
        select_idx = topk(covs, k=select_num, largest=True)
        # apply KL penalty of these samples
        pg_losses1[select_idx] += kl_coef * kl_penalty[select_idx]
        pg_loss = pg_losses1.mean()
   return pg_loss
```

Listing 1: The pseudo-code of the policy loss computation with Clip-Cov and KL-Cov. The implementation only need to modify several lines of code.

4.3 Experiments

Settings. We train Qwen2.5 models on math tasks to validate Clip-Cov and KL-Cov. We use the DAPO-MATH dataset (Yu et al., 2025) for training. In each rollout step, we sample 8 responses per prompt for a batch of 256 prompts using a temperature of 1, and subsequently perform 8 policy updates on the collected responses. We also filter out the prompts with all-correct/incorrect responses. The test datasets include MATH500, AIME 2024, AIME 2025 (Li et al., 2024), AMC, OMNI-MATH, OlympiadBench, and Minerva (Lewkowycz et al., 2022). During evaluation, we set the rollout temperature to 0.6 for AIME and AMC, while using greedy decoding for all other test sets. For baselines, we compare the original GRPO, and GRPO with Clip-higher, which tunes the upper threshold ϵ in PPO loss to 0.28 (Yu et al., 2025). In Clip-Cov, the clip ratio r is 2×10^{-4} , with ω_{low} and ω_{high} equals 1 and 5, respectively. For KL-Cov, the k is set as 2×10^{-3} and 2×10^{-4} for Qwen2.5-7B and 32B, respectively, the KL coefficient β is set as 1. The max generation length is 8192.

Results and analysis. We present the experimental results in Table 2, one can see that our two approaches both achieve non-trivial improvements across all benchmarks. Compared to GRPO, our method outperforms it by 2.0% on average for the 7B model and by 6.4% for the 32B model.

As shown in Figure 11, our method is able to maintain a considerably higher level of entropy throughout training. For example, when the baseline's entropy reaches a plateau and can no longer be consumed, the KL-Cov method still sustains an entropy level over $10 \times$ higher. Meanwhile, the response length of the policy model steadily increases, and its performance on the test set consistently surpasses that of the baseline. This indicates that our model is able to explore more "freely" during training, learning better policy through RL.

Compared to the clip-higher technique, although it can also increase entropy and lead to performance improvement in the early stage of training, it gradually becomes unstable, with performance saturating and declining. In contrast, our method obtains more stable entropy curves throughout training, ultimately achieving non-trivial improvements over the baselines.

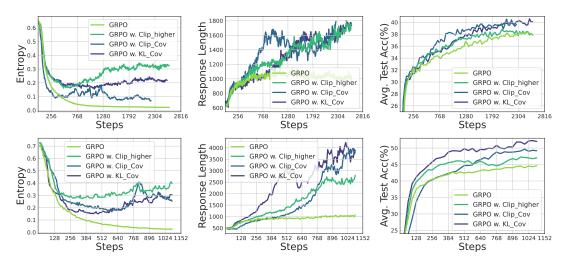


Figure 11: Training Qwen2.5-7B (*Top*) / Qwen2.5-32B (*bottom*) with GRPO with/without our methods. *Left:* Entropy dynamics. Our methods uplift policy entropy from collapse, enabling sustained exploration. *Middle:* Our method also incentivizes longer responses compared with vanilla GRPO. *Right:* The policy model consistently outperforms the baseline on testsets, avoiding performance plateaus.

Moreover, we observe that our method yields more substantial gains on the larger Qwen2.5-32B. Specifically, our method achieves improvements of **15.0%** and **14.6%** compared to GRPO on the most challenging benchmarks, AIME24 and AIME25, respectively. We infer that this is because the 32B model possesses greater potential from pretraining compared to the 7B model. Once the "exploration curse" caused by entropy collapse is lifted, the 32B model is able to explore more diverse and higher-quality policies.

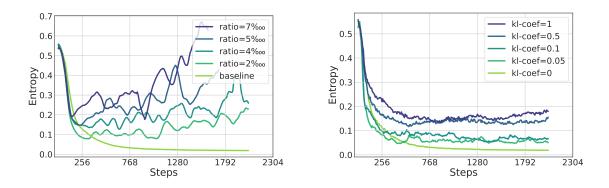


Figure 12: Differences in entropy dynamics of Qwen2.5-7B under varying KL coefficients and Clip ratios, evaluated Clip-Cov (*left*) and KL-Cov (*right*) settings, respectively.

4.4 Get Policy Entropy Controlled

We also evaluate the capability of our methods in controlling policy entropy as shown in Figure 12. For Clip-Cov, the level of policy entropy can be adjusted by tuning the ratio of clipped samples, where more clipped samples result in higher entropy. For KL-Cov, we can modulate the entropy by controlling the KL coefficient β , *i.e.*, the weight of the KL penalty. Specifically, a larger coefficient brings higher entropy. Comparing them, KL-Cov reaches stabler entropy curves than Clip-Cov, which might be preferable for stabilizing the training process. Although the optimal value of entropy under different scenarios remains an open question, our method demonstrates that we can simply tune the hyperparameters to control policy entropy, thus are capable of steering entropy and enabling the model to explore more effectively.

4.5 Discussion

Connection with clip-higher. Our main baseline, clip-higher (Yu et al., 2025), can also incentivize higher policy entropy. In fact, this technique shares similar functionality with our methods. By raising the upper threshold of the importance sampling ratio, clip-higher includes more low-probability tokens for policy updates. Also, the upper threshold only affects the tokens with positive advantages, which means clip-higher is actually adding more low-covariance (low probability, high advantage, with average covariance of \sim -0.03) tokens in gradient calculation. We take a step further by directly using the covariance as the threshold, thus controlling the entropy more precisely.

The philosophy of entropy control. In experiments, we find that the policy entropy is sensitive to hyperparameter settings. Specifically, our methods only interfere with a very small fraction of tokens $(10^{-4} \text{ to } 10^{-3})$, yet totally change the entropy curve. This means several "pivotal" tokens are crucial for the entropy of LLMs. Also, we don't observe a relationship between the intervened entropy and model performance. It still remains open whether there exists an optimal entropy value to balance the exploration and training stability.

5 Related Work

Policy entropy in reinforcement learning. Stemmed in information theory, entropy provides a principled mechanism to manage the exploitation-exploration tradeoff. Entropy-regularized reinforcement learning, also referred as maximum entropy RL (Ziebart et al., 2008; Toussaint, 2009), adopts a regularization term in reward to encourage high-entropy actions. This regularization term was widely-inherited in RL algorithms (Mnih et al., 2015, 2016; Schulman et al., 2017a,b; Haarnoja et al., 2017, 2018), and is viewed as a necessity. On the other hand, in RL for LLMs, there exist different opinions on whether entropy regularization should be preserved (Ouyang et al., 2022; Shao et al., 2024; Hu et al., 2025; He et al., 2025). Our experiments indicate that, it is necessary to control entropy, but we can design better objectives than entropy loss.

Predictability of reinforcement learning for reasoning language models. The first part of this work reveals the predictability of RL for LLM reasoning. The development of LLMs is largely guided by the neural scaling laws, which bridge model performances with computational budgets, model sizes, and the amount of training data (Hestness et al., 2017; Kaplan et al., 2020; Hoffmann et al., 2022). With scaling experiments on smaller models, the loss and task performance of larger models could be accurately predicted. In RL, Hilton et al. (2023); Rybkin et al. (2025) studied the scaling behavior of policy performances versus computing on non-LLM models, but the predictability of RL for LLMs has yet to be investigated. Gao et al. (2022) proposed to predict reward scores from KL divergence in RL on LLMs, which was used for modeling overoptimization effect of a proxy reward model. This work aligns with our conclusion considering that, 1) the verifiable reward eliminates the gap between the proxy reward model and ground truth; 2) the similarity between KL divergence and policy entropy.

Reinforcement learning for LLMs. Reinforcement learning has emerged as a major approach for LLM post-training (Ouyang et al., 2022; Meta, 2024; Team et al., 2023; Qwen et al., 2025; Jiang et al., 2023). Recent works have achieved further breakthrough on enhancing the reasoning capability of LLMs using RL with verifiable rewards (OpenAI, 2024a; Lambert et al., 2024; DeepSeek-AI et al., 2025; Team et al., 2025), drawing great attention in research community (Cui et al., 2025; Liu et al., 2025; Hu et al., 2025; He et al., 2025). However, there still lacks systematic study on the underlying mechanisms of RL for LLMs, which constitutes the primary goal of our work.

6 Conclusion

In this study, we try to address the challenge of policy entropy collapse in reinforcement learning for large language model reasoning. We empirically demonstrate that performance gains are often achieved by sacrificing exploratory capacity, which in turn imposes a foreseeable limit on improvement. To gain a deeper understanding, we conduct a theoretical investigation into entropy dynamics and introduce two simple regularization techniques, Clip-Cov and KL-Cov, to directly manage high-covariance tokens and thereby counteract entropy collapse. Looking further, RL has been identified as the next scaling axis after pre-training. However, scaling computing for RL requires more than entropy minimization. We hope this research could provide valuable insights into the role of entropy, fostering RL to reach a higher level of intelligence.

References

- Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. <u>Journal of Machine Learning Research</u>, 22(98): 1–76, 2021.
- Shivam Agarwal, Zimin Zhang, Lifan Yuan, Jiawei Han, and Hao Peng. The unreasonable effectiveness of entropy minimization in llm reasoning. arXiv preprint arXiv:2505.15134, 2025.
- Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms. arXiv preprint arXiv:2402.14740, 2024.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. arXiv preprint arXiv:2110.14168, 2021.
- Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards, 2025. URL https://arxiv.org/abs/2502.01456.
- DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Oinyu Chen, Oiushi Du, Ruigi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.
- Benjamin Eysenbach and Sergey Levine. Maximum entropy rl (provably) solves some robust rl problems. arXiv preprint arXiv:2103.06257, 2021.
- Bofei Gao, Feifan Song, Zhe Yang, Zefan Cai, Yibo Miao, Qingxiu Dong, Lei Li, Chenghao Ma, Liang Chen, Runxin Xu, et al. Omni-math: A universal olympiad level mathematic benchmark for large language models. arXiv preprint arXiv:2410.07985, 2024.
- Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In <u>International</u> Conference on Machine Learning, 2022.
- Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In International conference on machine learning, pp. 1352–1361. PMLR, 2017.

- Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In <u>International conference on machine</u> learning, pp. 1861–1870. Pmlr, 2018.
- Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 3828–3850, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.
- Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reaonser series. https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reaonser-Series-1d0bc9ae823a80459b46c149e4f51680, 2025. Notion Blog.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. <u>arXiv:preprint</u> arXiv:2103.03874, 2021.
- Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. arXiv preprint arXiv:1712.00409, 2017.
- Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. <u>arXiv</u> preprint arXiv:2301.13442, 2023.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. arXiv preprint arXiv:2203.15556, 2022.
- Jian Hu. Reinforce++: A simple and efficient approach for aligning large language models. <u>arXiv preprint</u> <u>arXiv:2501.03262</u>, 2025.
- Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model. <u>arXiv:preprint arXiv:2503.24290</u>, 2025.
- Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.
- Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022.
- Sham M Kakade. A natural policy gradient. Advances in neural information processing systems, 14, 2001.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. <u>arXiv:2001.08361</u>, 2020.
- Wouter Kool, Herke van Hoof, and Max Welling. Buy 4 reinforce samples, get a baseline for free! In DeepRLStructPred@ICLR, 2019. URL https://api.semanticscholar.org/CorpusID:198489118.
- Nathan Lambert, Jacob Daniel Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hanna Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. ArXiv, abs/2411.15124, 2024.

- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. Advances in Neural Information Processing Systems, 35:3843–3857, 2022.
- Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, et al. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. Hugging Face repository, 13:9, 2024.
- Jiacai Liu. How does rl policy entropy converge during iteration? https://zhuanlan.zhihu.com/p/28476703733, 2025. URL https://zhuanlan.zhihu.com/p/28476703733.
- Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. arXiv preprint arXiv:2503.20783, 2025.
- Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Tianjun Zhang, Li Erran Li, et al. Deepscaler: Surpassing o1-preview with a 1.5 b model by scaling rl. Notion Blog, 2025.
- Meta. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.
- Meta-Llama-3.2. Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices.
- MistralAI-NeMo. Mistralai-nemo. URL https://mistral.ai/news/mistral-nemo.
- MistralAI-Small-3. Mistralai-small-3. URL https://mistral.ai/news/mistral-small-3.
- Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529–533, 2015.
- Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pp. 1928–1937. PmLR, 2016.
- OpenAI. Openai o1 system card. ArXiv, abs/2412.16720, 2024a.
- OpenAI. Gpt-4 technical report, 2024b. URL https://arxiv.org/abs/2303.08774.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. Advances in neural information processing systems, 35:27730–27744, 2022.
- Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.
- Oleh Rybkin, Michal Nauman, Preston Fu, Charlie Snell, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Value-based deep rl scales predictably. arXiv preprint arXiv:2502.04327, 2025.
- John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International conference on machine learning, pp. 1889–1897. PMLR, 2015.
- John Schulman, Xi Chen, and Pieter Abbeel. Equivalence between policy gradients and soft q-learning. <u>arXiv:1704.06440</u>, 2017a.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017b.

- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.
- Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. <u>arXiv preprint arXiv: 2409.19256</u>, 2024.
- David Silver and Richard S Sutton. Welcome to the era of experience. Google AI, 2025.
- Richard Sutton. The bitter lesson. Incomplete Ideas (blog), 13(1):38, 2019.
- Richard S Sutton. Learning to predict by the methods of temporal differences. <u>Machine learning</u>, 3:9–44, 1988.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805, 2023.
- Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, et al. Kimi k1. 5: Scaling reinforcement learning with llms. <u>arXiv:2501.12599</u>, 2025.
- Marc Toussaint. Robot trajectory optimization using approximate inference. In <u>Proceedings of the 26th annual</u> international conference on machine learning, pp. 1049–1056, 2009.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, 8:229–256, 1992.
- Ronald J Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. Connection Science, 3(3):241–268, 1991.
- Zhangchen Xu, Yang Liu, Yueqin Yin, Mingyuan Zhou, and Radha Poovendran. Kodcode: A diverse, challenging, and verifiable synthetic dataset for coding, 2025. URL https://arxiv.org/abs/2503.02951.
- Jianhao Yan, Yafu Li, Zican Hu, Zhi Wang, Ganqu Cui, Xiaoye Qu, Yu Cheng, and Yue Zhang. Learning to reason under off-policy guidance. arXiv:2504.14945, 2025.
- Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Honglin Yu, Weinan Dai, Yuxuan Song, Xiang Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yong-Xu Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. 2025. URL https://api.semanticscholar.org/CorpusID:277104124.
- Lifan Yuan, Wendi Li, Huayu Chen, Ganqu Cui, Ning Ding, Kaiyan Zhang, Bowen Zhou, Zhiyuan Liu, and Hao Peng. Free process rewards without process labels. <u>International conference on machine learning</u>, 2025.
- Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? <u>ArXiv</u>, abs/2504.13837, 2025.
- Huaye Zeng, Dongfu Jiang, Haozhe Wang, Ping Nie, Xiaotong Chen, and Wenhu Chen. Acecoder: Acing coder rl via automated test-case synthesis, 2025. URL https://arxiv.org/abs/2502.01718.
- Qingyang Zhang, Haitao Wu, Changqing Zhang, Peilin Zhao, and Yatao Bian. Right question is already half the answer: Fully unsupervised llm reasoning incentivization. arXiv preprint arXiv:2504.05812, 2025.
- Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In Aaai, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.
- Yuxin Zuo, Kaiyan Zhang, Shang Qu, Li Sheng, Xuekai Zhu, Biqing Qi, Youbang Sun, Ganqu Cui, Ning Ding, and Bowen Zhou. Ttrl: Test-time reinforcement learning. arXiv preprint arXiv:2504.16084, 2025.

A Training Details for Different Models

Due to inherent differences in the initial reasoning abilities between model families, we train models using data of different difficulty levels to stabilize the RL process Specifically, for math tasks, we train the Qwen family and Mistral-24B model using Eurus-2-RL-Math (Cui et al., 2025), while other model families are trained using GSM8K (Cobbe et al., 2021). The downstream performance is evaluated using MATH500 (Hendrycks et al., 2021), AIME 2024 (Li et al., 2024), AMC (Li et al., 2024), OlympiadBench (He et al., 2024), and OMNI-MATH (Gao et al., 2024). For code tasks, we train the Qwen family and Mistral-24B model using AceCode (Zeng et al., 2025), Eurus-2-RL-Code (Cui et al., 2025), and Kodcode³.

B More Fitting Results

In this section, we present more fitting experiment results.

C Fitting Results of Training with Different Dataset.

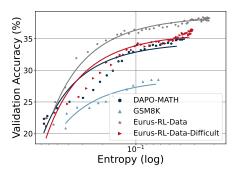


Figure 13: Training Qwen2.5-7B with different data.

D Fitting Results of Instruct Models

We also conduct fitting experiments on instruct models, and the fitting function remains valid in our experiments. We present the fitting results here.

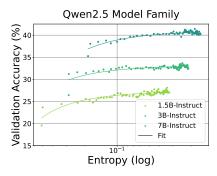


Figure 14: Training Qwen2.5 instruct models on math task.

³We process the data with style instruct and complete into a format that can be handled by unit tests. For the online-judge style, we removed this portion of the data as it was derived from instruct style data.

E Proof

E.1 Useful Lemmas

Lemma 2 (Derivative of softmax function)

$$\frac{\partial \log \pi_{\theta}(a \mid s)}{\partial \theta_{s,a'}} = \mathbf{1} \left\{ a = a' \right\} - \pi_{\theta} \left(a' \mid s \right)$$

Lemma 3 (Expectation of Advantage function given state s)

$$\mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} [A^{\pi_{\theta}}(s, a)] = \mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} [Q^{\pi_{\theta}}(s, a) - V^{\pi_{\theta}}(s)]$$

$$= \mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} [Q(s, a)] - \mathbb{E}_{a \sim \pi_{\theta}(\cdot|s)} [V(s)]$$

$$= V(s) - V(s)$$

$$= 0$$

E.2 Proof for Lemma 1

Lemma 1: Let the actor policy π_{θ} be a tabular softmax policy, the difference of information entropy given state s between two consecutive steps satisfies

$$\mathcal{H}(\pi_{\theta}^{k+1}|s) - \mathcal{H}(\pi_{\theta}^{k}|s) \approx -\text{Cov}_{a \sim \pi_{\theta}^{k}(\cdot|s)} \left(\log \pi_{\theta}^{k}(a|s) \;, z_{s,a}^{k+1} - z_{s,a}^{k}\right)$$

Proof adapted from (Liu, 2025).

In tabular softmax policy, each state-action pair (s,a) is associated with an individual logit parameter $z_{s,a} = \theta_{s,a}$. We assume that we are updating logits z via $z^{k+1} = z^k + \eta \cdot \nabla J(\pi_\theta)$. Given η is relatively small, leveraging Taylor's expansion under first-order approximation, we have

$$\mathcal{H}(\pi_{\theta}^{k+1} \mid s) \approx \mathcal{H}(\pi_{\theta}^{k} \mid s) + \langle \nabla \mathcal{H}(\pi_{\theta}^{k} \mid s), (z^{k+1} - z^{k}) \rangle$$

We then to derive what $\nabla \mathcal{H}(\pi_{\theta}^k \mid s)$ is, according to the definition of \mathcal{H} , we have

$$\nabla_{\theta} \mathcal{H}(\pi_{\theta} \mid s) = \nabla_{\theta} \mathcal{H}(\pi_{\theta}(\cdot \mid s))$$

$$= \nabla_{\theta}(-\mathbb{E}_{a \sim \pi_{\theta}(\cdot \mid s)} [\log \pi_{\theta}(a \mid s)])$$

$$= -\mathbb{E}_{a \sim \pi_{\theta}(\cdot \mid s)} [\nabla_{\theta} \log \pi_{\theta}(a \mid s) + \log \pi_{\theta}(a \mid s)\nabla_{\theta} \log \pi_{\theta}(a \mid s)]$$

$$= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} [\log \pi_{\theta}(a \mid s)\nabla_{\theta} \log \pi_{\theta}(a \mid s)]$$

Then we have,

$$\begin{split} \langle \nabla_{\theta} \mathcal{H}(\theta^{k} \mid s), (z^{k+1} - z^{k}) \rangle &= -\langle \mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \nabla_{\theta} \log \pi_{\theta}(a \mid s) \right], (\theta^{k+1} - \theta^{k}) \rangle \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \langle \nabla_{\theta} \log_{\pi_{\theta}}(a \mid s), \theta^{k+1} - \theta^{k} \rangle \right] \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \sum_{a' \in \mathcal{A}} \frac{\partial \log \pi_{\theta}(a \mid s)}{\partial \theta_{s,a'}} \cdot (\theta^{k+1}_{s,a'} - \theta^{k}_{s,a'}) \right] \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \sum_{a' \in \mathcal{A}} \left(\mathbf{1} \left\{ a = a' \right\} - \pi \left(a' \mid s \right) \right) \cdot (\theta^{k+1}_{s,a'} - \theta^{k}_{s,a'}) \right] \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \left[(\theta^{k+1}_{s,a} - \theta^{k}_{s,a}) - \sum_{a' \in \mathcal{A}} \pi(a' \mid s) (\theta^{k+1}_{s,a'} - \theta^{k}_{s,a'}) \right] \right] \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) (\theta^{k+1}_{s,a} - \theta^{k}_{s,a}) \right] + \mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \cdot \mathbb{E}_{a' \sim \pi(\cdot \mid s)} \left[\theta^{k+1}_{s,a'} - \theta^{k}_{s,a'} \right] \right] \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) (\theta^{k+1}_{s,a} - \theta^{k}_{s,a}) \right] + \mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \cdot \mathbb{E}_{a' \sim \pi(\cdot \mid s)} \left[\theta^{k+1}_{s,a'} - \theta^{k}_{s,a'} \right] \right] \\ &= -\mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) (\theta^{k+1}_{s,a} - \theta^{k}_{s,a}) \right] + \mathbb{E}_{a \sim \pi(\cdot \mid s)} \left[\log \pi_{\theta}(a \mid s) \cdot \mathbb{E}_{a' \sim \pi(\cdot \mid s)} \left[\theta^{k+1}_{s,a'} - \theta^{k}_{s,a'} \right] \right] \\ &= -Cov_{a \sim \pi(\cdot \mid s)} \left(\log \pi(a \mid s), \theta^{k+1} - \theta^{k} \right) \\ &= -Cov_{a \sim \pi(\cdot \mid s)} \left(\log \pi(a \mid s), z^{k+1} - z^{k} \right) \end{aligned}$$

E.3 Proof for Proposition 1

Proposition 1: Let the actor policy π_{θ} be tabular softmax policy and updated using Eq. 2, the difference of $z_{s,a}$ between two consecutive steps satisfies

$$z_{s,a}^{k+1} - z_{s,a}^{k} = \eta \cdot \pi_{\theta}(a \mid s) \cdot A(s,a)$$

Proof.

In tabular softmax policy, each state-action pair (s,a) is associated with an individual logit parameter $z_{s,a} = \theta_{s,a}$. Through gradient backtracking, $z_{s,a}$ is updated via $z_{s,a}^{k+1} = z_{s,a}^k + \eta \cdot \nabla_{\theta_{s,a}} J(\theta)$, therefore, we have

$$\begin{split} z_{s,a}^{k+1} - z_{s,a}^{k} &= \eta \cdot \nabla_{\theta_{s,a}} J(\theta) \\ &= \eta \cdot \underset{a' \sim \pi_{\theta}(\cdot \mid s)}{\mathbb{E}} \left[\nabla_{\theta_{s,a}} \log \pi_{\theta}(a' \mid s) \cdot A(s, a') \right] \\ &= \eta \cdot \underset{a' \sim \pi_{\theta}(\cdot \mid s)}{\mathbb{E}} \left[\underbrace{\frac{\partial \log \pi_{\theta}(a' \mid s)}{\partial \theta_{s,a}} \cdot A(s, a')}_{\text{Lemma 2}} \cdot A(s, a') \right] \\ &= \eta \cdot \sum_{a' \in \mathcal{A}} \left[\pi_{\theta}(a' \mid s) \cdot (\mathbf{1} \left\{ a = a' \right\} - \pi_{\theta}(a \mid s)) \cdot A(s, a') \right] \\ &= \eta \cdot \pi_{\theta}(a \mid s) \cdot \left[(1 - \pi_{\theta}(a \mid s)) \cdot A(s, a) - \sum_{a' \in \mathcal{A}, a' \neq a} \pi_{\theta}(a' \mid s) \cdot A(s, a') \right] \\ &= \eta \cdot \pi_{\theta}(a \mid s) \cdot \left[A(s, a) - \sum_{\underline{a' \in \mathcal{A}}} \pi_{\theta}(a' \mid s) \cdot A(s, a') \right] \\ &= \eta \cdot \pi_{\theta}(a \mid s) \cdot [A(s, a) - 0] \\ &= \eta \cdot \pi_{\theta}(a \mid s) \cdot A(s, a) \end{split}$$

E.4 Proof for Theorem 2

Theorem 2: Let the actor policy π_{θ} be tabular softmax policy, and π_{θ} is updated via natural policy gradient Kakade (2001), the difference of information entropy given state s between two consecutive steps satisfies

$$\mathcal{H}(\pi_{\theta}^{k+1}|s) - \mathcal{H}(\pi_{\theta}^{k}|s) \approx -\eta \cdot \text{Cov}_{a \sim \pi_{\theta}^{k}(\cdot|s)} \left(\log \pi_{\theta}^{k}(a|s) \;, A(s,a)\right)$$

Proof.

According to Lemma 1, we first derive the difference of logits z in natural policy gradient. We learn from (Agarwal et al., 2021) that, when we are updating policy using natural policy gradient via gradient backtracking, $z_{s,a}^{k+1} - z_{s,a}^k$ satisfies,

$$z_{s,a}^{k+1} - z_{s,a}^{k} = \eta \cdot A(s,a)$$

Applying this into Lemma 1, we have

$$\mathcal{H}(\pi_{\theta}^{k+1}|s) - \mathcal{H}(\pi_{\theta}^{k}|s) \approx -\eta \cdot \mathsf{Cov}_{a \sim \pi_{\theta}^{k}(\cdot|s)} \left(\log \pi_{\theta}^{k}(a|s) \;, A(s,a)\right)$$